

THE PATENTS ACT, 1970

Jc971 U.S. PTO
10/083174
02/25/02

IT IS HEREBY CERTIFIED THAT, the annex is a true copy of Application and Provisional specification filed on 6.8.2001 in respect of Patent Application No. 760/Mum/2001 of Tata Consultancy Services(a Division of Tata Sons Limited) of Bombay House, Sir Momi Mody Street, Mumbai-400 023, Maharashtra, India an Indian Company.

This certificate is issued under the powers vested on me under Section 147(1) of the Patents Act, 1970.....

.....Dated this 25th day of January 2002.

N. K. Garg
(N.K.Garg)

Asst Controller of Patents & Designs

CERTIFIED COPY OF
PRIORITY DOCUMENT

THIS PAGE BLANK (USPTO)

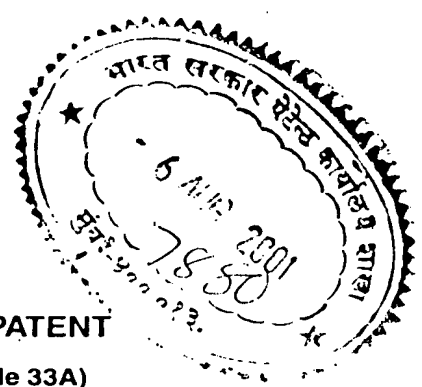
FORM-1

THE PATENTS ACT, 1970

(39 OF 1970)

APPLICATION FOR GRANT OF A PATENT

(See sections 5(2), 7, 54 and 135 and rule 33A)



1. We, TATA CONSULTANCY SERVICES (a Division of TATA SONS LIMITED),
of Bombay House, Sir Homi Mody Street, Mumbai 400 023, Maharashtra,
India, an Indian Company,



2. hereby declare :-

(a) that I am/we are in possession is an invention titled AN APPARATUS AND METHOD
FOR IMPLEMENTING BATCH PROGRAMS

(b) that the Provisional/Complete specification relating to this invention is filed with this
application

(c) that there is no lawful ground of objection to the grant of a patent to me/us.

3. further declare that the inventor(s) for the said invention is/are :

(a) VINAY KULKARNI of Tata Consultancy Services, Hadapsar Industrial
Estate, Pune 411 013, Maharashtra, India, an Indian National;

(b) VILAS PRABHU of Tata Consultancy Services, Hadapsar Industrial
Estate, Pune 411 013, Maharashtra, India, an Indian National;

(c) SREEDHAR REDDY of Tata Consultancy Services, Hadapsar Industrial
Estate, Pune 411 013, Maharashtra, India, an Indian National; and

(d) SHASHANK KULKARNI of Tata Consultancy Services, Hadapsar Industrial
Estate, Pune 411 013, Maharashtra, India, an Indian National.

760/11407/2001
6/8/2001

Received by	5264	In Cash
Received by	6/8/01	
Vide Entry	3191	In the
Register of	Patents	

ORIGINAL

Electronic

cash

6/8/01

4. I/We, claim the priority from the application(s) filed in convention countries, particulars of which are as follows :

(a) [Country]

(b) [Appln.No.]

(c) [Date of Appln.]

N.A.

(d) [Applicant in Convention Country]

(e) [Title]

5. I/We state that the said invention is an improvement in or modification of the invention, the particulars of which are as follows and of which I/We are the applicant/patentee :

(a) Application No.

(b) Date of application

N.A.

6. I/We state that the application is divided out of my/our application, the particulars of which are given below and pray that this application deemed to have been filed on _____ date under section 16 of the Act.

(a) Application No.

(b) Date of filing of provisional specification :

N.A.

and date of filing of complete specification :

7. That ~~am~~ I/We are the assignee or legal representative of the true and first inventors.

8. That ~~my~~ our address for service in India is as follows : **R.K. DEWAN & COMPANY**, Trade Marks & Patents Attorneys, 78, Podar Chambers, S.A.Brelvi Road, Fort, Mumbai 400 001, Maharashtra, India, Tel. (91) 22-2661662/2663002, Telefacsimile number (91) 22-2650159, Email>rkdeewan@vsnl.com.



CONVENTION COUNTRY :

~~Vinay~~
(Vinay Kulkarni) Vilas Prabhu. Sreedhar ~~Shreshant~~
(Vilas Prabhu) (Sreedhar Reddy) (Shreshant Kulkarni)

11. Following are the attachment with the application :

(a) Provisional/~~Complete~~ specification (3 copies)

(b) Drawings (3 Copies)

(c) ~~Priority document(s)~~

(d) ~~Statement of Invention~~
Copy of General

(e) Power of authority

(f)

(g)

(h)

(i) Fee Rs.

In cash/cheque/bank draft bearing No.

date

on

Bank

We request that a patent may be granted to me/us for the said invention.

Dated this 31st day of July 2001.

Signature :

Hemant

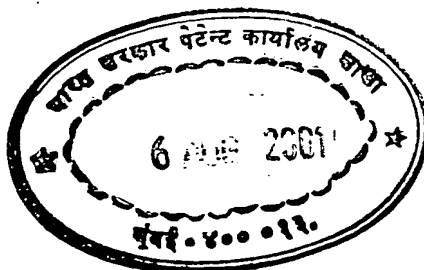
Name : (HEMANT DHANSUKHLAL VAKIL)

To :

The Controller of Patents

The Patent Office

at MUMBAI



FORM-2

THE PATENT ACT, 1970

PROVISIONAL

Specification

SECTION - 10

AN APPARATUS AND METHOD FOR IMPLEMENTING
BATCH PROGRAMS

TATA CONSULTANCY SERVICES,
(a Division of TATA SONS LIMITED),
of Bombay House, Sir Homi Mody Street,
Mumbai 400 023, Maharashtra, India,
an Indian Company

THE FOLLOWING SPECIFICATION DESCRIBES
THE NATURE OF THIS INVENTION :-

760 | मुंबई | 2001
MUM

6 AUG 2001

ORIGINAL

This invention relates to an apparatus and method for Implementing Batch Programs

Typically, a batch program has the following characteristics,

- Processing of large volume of data
- Resource intensive nature
- Ability to recover from errors and restart with minimal loss of computations performed so far
- Must finish within a given time-frame

Designing and developing a batch program is an involved task as several orthogonal strategies need to be integrated in a consistent and homogenous manner. The issues involved are,

- Memory management
- Recovery and restart
- Scheduling
- Error logging and diagnostics

In the prior art there is no method or apparatus for generating and managing batch programs in a comprehensive manner.

The salient object of the present invention include

To provide a comprehensive support for memory management, recovery and restart and Error logging and diagnostics

- To provide a Model based generative approach leading to increased productivity
- To provide a Framework based solution enabling customization

The invention will now be described with reference to the accompanying drawings, in which

Figure 1 shows a model for the proposed batch architecture.

Figure 2 shows the structure of a batch program;

Figure 3 illustrates a schematic drawing of a batch function in accordance with the present invention;

Figure 4 illustrates a batch function in accordance with the [resent invention;

Figure 5 illustrate framework classes in accordacne with this invention;

Figure 6 shows how Every modelled batch program is realised as a class in accordance with this invention ;

Figure 7 shows the Function main for every modeled batch program generated;

Figure 8, shows how a batch function model is realised as a class ;

Figure 9. shows the generated code for a shell of a batch function ;

Figure 10 shows a code sample how a framework manages memory for its invocation;

Figure 11 shows how all modeled classes inherit from class CppObj which provides the behaviour; and

Figure 12 shows the Template for Set<attributeName> method for every modelled class.

Referring to the drawings, Fig 1 shows a model for the proposed batch architecture. The salient features of the model shown in Figure 1 are,

- BatchPgm denotes an executable program which can be scheduled using a batch scheduler.
- A batch program is a set of batch functions being invoked in a sequence as well as in a nested manner.
- A batch function is characterised by,
- A data provider providing the data to be processed by Process function
- Pre, Process and Post operations (as an Operation in UML),
- Context class encapsulating the restart information as well as the information to be passed between Pre, Process and Post operations
- A property to indicate if memory management for this batch function should be provided by the framework

The above associations are optional.

- A batch program has a starting batch function associated with it. It is the first batch function to be invoked when program control reaches the batch program.

2.1.1 Structure of a batch program

The batch architecture model defines the structure of a batch program. Model based generative approach provides an automated mechanism for realising this structure. Framework supported realisation renders flexibility to the structure.

A batch program is an independent process.

A batch program conforms to the structure seen in Figure 3 of the accompanying drawings,

Step1: Perform system initialisations

Step2: Perform set up required for this batch program to be invoked

Step3: Invoke starting batch function of this batch program by passing the necessary context information

Step4: Log errors, if any.

Step5: Perform wind up activities for this batch

Step6: Release system resources obtained, if any.

Step7: Return with the status

The framework provides a default behaviour for steps 1, 4, 5 and 6 above. User can override the default behaviour.

2.1.2 Structure of a batch function

A batch function is invoked either through a batch program (as its starting function) or through Pre-, Process- or Post- operations of another batch function of the batch program as shown schematically in figure 2

A batch function accepts restartFlag and batch program context as input parameters, batch function context as in-out parameter and returns status. A batch function conforms to the structure as seen Figure 4 of the accompanying drawings,

The structure seen in Figure 4 reflects purely the structure of a batch function. Behaviour of a batch function is provided by Pre, Post and Process operations and is completely under the control of user. Nesting of batch functions is possible by invoking a batch function (callee) from the Process operation of a batch function (caller). Sequencing of batch functions is possible by invoking a batch function (next) from the Post operation of a batch function (current) or a batch function (prev) from the Pre operation of a batch function (current).

Framework provides default behaviour for Commit() which can be overridden. Framework provides a modeling abstraction for data provider and a generation mechanism for the same.

The data provider is responsible for providing the data to be processed by a batch function. Typically, the data is provided by a database query in chunks using cursor. Output of data provider is passed as input to the Process operation of a batch function. Data provider is expected to make use of the context class of the batch function so as to provide correct chunk of data in normal as well as restart mode of operation.

Context class of a batch function encapsulates the following,

- Restart information for the batch function
- Restart data for the data provider
- Data required for Pre, Post and Process operations to communicate with each other

Framework ensures that context of a batch function is written to persistent storage at the time of commit.

Framework classes provide a default behaviour for commit control based on commit frequency. These classes can be overridden to provide customised commit control as explained later.

Batch architecture is supported through two public classes viz., BatchPgm and BatchFunction, and a persistent table BatchControlTable containing < BatchProgramId, RestartStatus, BatchContext_in_serialised_form >.

Framework classes are illustrated in figure 5 of the accompanying drawings

Every modelled batch program is realised as a class as shown in figure 6 of the accompanying drawings ,

Function main for every modeled batch program is generated as shown in Figure 7,

Every batch function modelled is realised as a class as shown in Figure 8,

Generated code for a shell of a batch function is as shown in Figure 9.

2.4 Memory management

A batch function can be tagged as *MemoryManaged* in the model. In which case, framework manages memory for its invocation as shown in the code sample as seen in Figure 10.

All modelled classes inherit from class CppObj which provides the behaviour as seen in Figure 11 of the accompanying drawings ,

Template for Set<attributeName> method for every modelled class is as seen in Figure 12 of the accompanying drawings,

Application errors encountered so far are collected, along with the application context, in a in-memory container object. The container class provides methods for logging the errors to a persistent storage. The framework calls these methods at commit and exit points. Framework provides a default implementation which can be overridden by user.

The Framework as envisaged in accordance with this invention supports multi-threading through data dependent routing. The data to be processed is divided into multiple (say N) orthogonal sets and this separation of data is apriori (and hence possible to code in a data provider of a batch function). Thus, multi-threading like behaviour is achieved by invoking the N batch programs simultaneously through a batch scheduler.

The framework allows the following behaviours to be customised:

- Initialisation of a batch program through overridden `<BatchProgramName>::Init()`
- Commit behaviour of a batch program through overridden `<BatchProgramName>::Commit()` and `<BatchProgramName>::ResetCommitInfo()`
- Commit behaviour of a batch function through overridden `<BatchFunctionName>::Commit()` and `<BatchFunctionName>::ResetCommitInfo()`
- Class `BatchProgram` provides a shared area, in the form of `Darray` `commitArray`, for co-ordinating commits between a batch program and its batch functions
- Error logging behaviour through overridden `<BatchProgramName>::LogErrors()` method. Framework supplies the errors encountered in a global `Darray`.
- External world can call batch functions through `BatchPgm::CallBatchFunction()`.

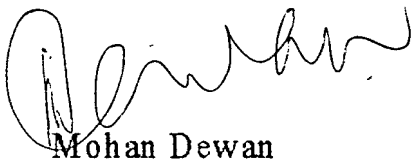
The invention envisaged in accordance with this invention ,

- Provides tool support for generating and managing batch programs in a comprehensive manner and automates it to a large extent.
- Provides a higher level of abstraction
- Provides increased productivity due to model based generative approach

- Addresses the complex issues such as Memory management, Recovery and restart, Scheduling, and Error logging by providing a default behaviour
- Enables customisability due to framework based approach
- Delivers a platform-agnostic solution which can be easily realised on other equivalent platforms

Although the invention has been described in terms of particular embodiments and applications, one of ordinary skill in the art, in light of this teaching, can generate additional embodiments and modifications without departing from the spirit of or exceeding the scope of the invention. Accordingly, it is to be understood that the drawings and descriptions herein are proffered by way of example to facilitate comprehension of the invention and should not be construed to limit the scope thereof.

Dated this 31st day of July 2001



Mohan Dewan

Of R. K. Dewan & Co.,

Applicants' Patent Attorney

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 1

NO. : /MUM/01

PROVISIONAL SPECIFICATION

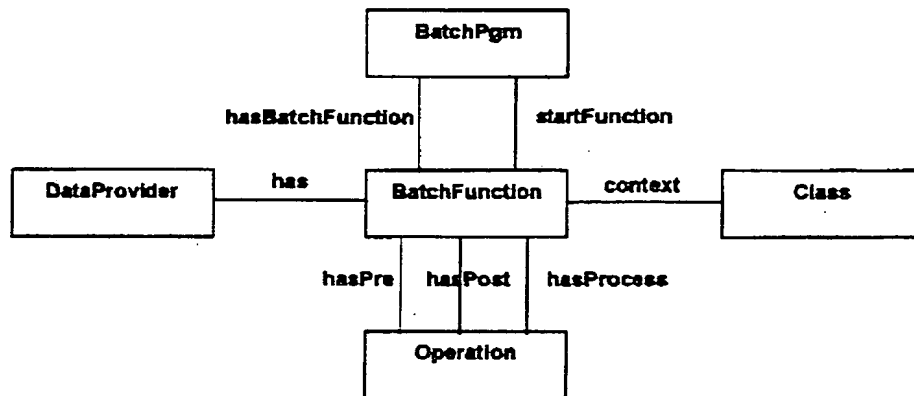


Figure 1. Model for Batch Architecture

MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

- 6 AUG 2001

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 2

NO. : /MUM/01

PROVISIONAL SPECIFICATION

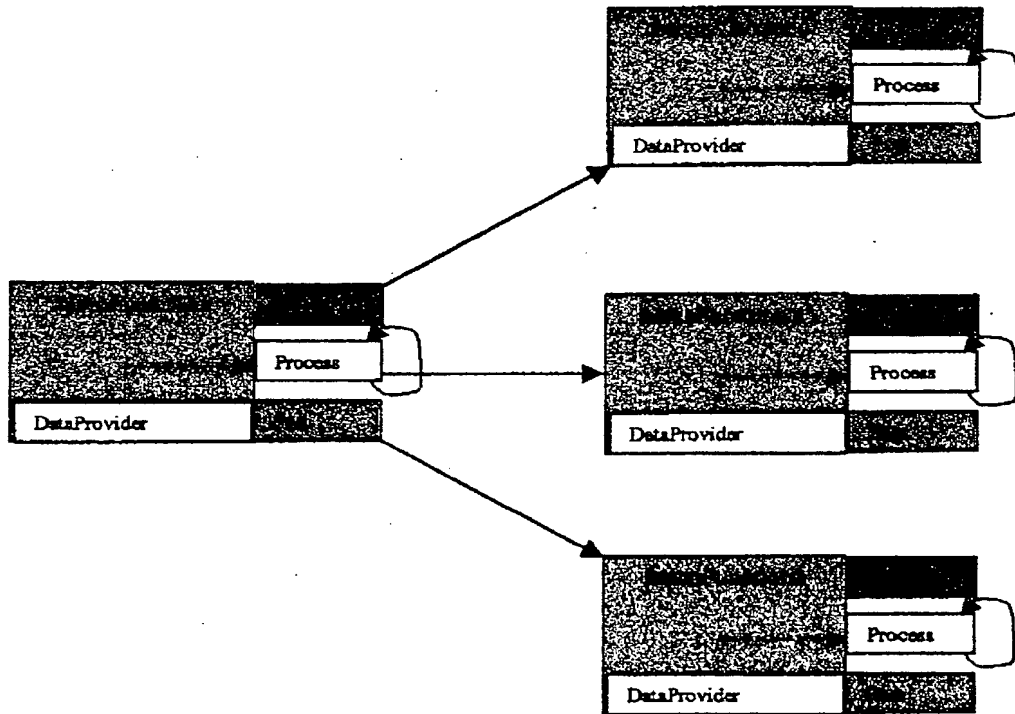


Figure 2. Structure of a batch program

```
Int main( int argc, char *argv)
{
```

Figure 3

MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

2-6 AUG 2001

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 3

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
If ( restart )
Then
    Obtain batch function context at the time of last successful
    commit.
    If ( previous failure was in Pre function ) Then goto Pre.
    If ( previous failure was in Process function ) Then goto
    Process.
    If ( previous failure was in Post function ) Then goto Post.
Else
    Obtain batch function context from the input parameter.
Fi
Pre:
    Push context of this batch function onto stack
    Invoke Pre operation of the batch function by passing the necessary
    parameters.
    Pop the stack

Process:
    For all objects fetched by the associated data provider
    Do
        Push context of this batch function onto stack
        Invoke Process operation of the batch function by passing the
        necessary parameters.
        Invoke Commit operation of the framework
        Pop the stack
    Done

Post:
    Push context of this batch function onto stack
    Invoke Post operation of the batch function by passing the necessary
    parameters.
    Pop the stack
```

Figure 4



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 4

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
typedef ErrorStatus (*BatchFn) (BatchPgm*, CppObj*, CppObj*&);

Class BatchPgm
{
protected:
    long count;          // counter
    long commitThreshold; // commit frequency
    BatchFn startingBatchFunction;
    char *batchPgmId;
    long restartStatus;

    // context of starting batch function
    CppObj *startFuncContext;

    // Stack of Batch Function contexts
    XLDArrary *bfRestartStack;

    // Stack of Batch Function contexts
    XLDArrary *bfContextStack;

    // Darray of < "<batchFunctionName>", void * >
    // i.e. struct CommitControl used by overriding
    // Commit() functions
    // void * serves as shared memory using which BatchProgram and
    // BatchFunction coordinate commit behaviour. For example, the
    // data could be <batchFunctionName> instance in a serialised
    // form
    Darray *commitArray;

    // Array to hold
    // <batchFunctionName, batchFunction, memMgmtFlag, BatchFunction>
    // tuples for the batch program
    static BatchFunctionDescriptor batchFunctionArray[];
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 5

NO. : /MUM/01

PROVISIONAL SPECIFICATION

Public:

```
// Constructor
<BatchProgram> ( char *batchProgramId, BatchFn startFunc ) :
batchPgmId(batchProgramId), startingBatchFunction(startFunc)
{
    bfContextStack = new XLDArrary;
    bfRestartStack = new XLDArrary;
    commitAry = new DArray;
    startFuncContext = NULL;
}

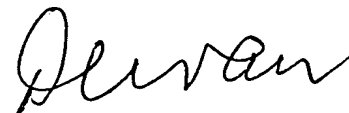
long GetRestartStatus() { return restartStatus; }

// To set up DB connections, initialise memory pools, message
// array etc. after invoking Init( argc, argv )
// Format of configuration file for a batch program
// BatchProgram <batchProgramName>
// {
//     commitThreshold = <value>;
//     batchPgmId = <value>;
//     DbConnectionInfo = <DbConnectionString>;
//
//     BatchFunction <batchFunctionName>
//     {
//         commitThreshold = <value>;
//     }
//     .
//     .
//     .
// }
```

```
ErrorStatus Setup( int argc, char *argv [] )
{
    // Parse input arguments
    // -c gives configuration file name for batch program
    char *configFileName;

    // Open config file and set the following
    // - BatchPgm::commitThreshold
    // - BatchPgm::batchPgmId
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 6

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
//      - for each BatchFunction specified
//          - Instantiate <BatchFunction> object
//          - Set commitThreshold
//          - Add to the corresp element in
//              batchFunctionArray
//      - for each BatchFunction not specified
//          - Instantiate <BatchFunction> object
//          - Set commitThreshold = 0
//          - Add to the corresp element in
//              batchFunctionArray
//
//      - Read DbConenctionInfo and connect to DB
//
// If config file does'nt exist or is empty or the field
// unspecified
//      - set BatchPgm::commitThreshold = 1
//      - for each element in batchFunctionArray
//          - Instantiate <BatchFunction> object
//          - Set commitThreshold = 0
//          - Add to the corresp element in
//              batchFunctionArray

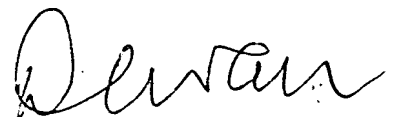
return Init( argc, argv );
}

// Complementary of Setup()
ErrorStatus Cleanup ();

// default Init() implementation supplied by framework
Virtual ErrorStatus Init( int argc, char *argv ){}

// Entry point of the batch program
ErrorStatus Exec()
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 7

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
{  
    ErrorStat retVal;  
    BatchControlInfo *batchControlInfo =  
        new BatchControlInfo( batchPgmId );  
  
    retVal = batchControlInfo->Get();  
    if( retVal != DM_SUCCESS )  
    {  
        batchControlInfo->SetRestartStatus( FRESH );  
        restartStatus = FRESH;  
        retVal = batchControlInfo->Create();  
        if( retVal != DM_SUCCESS )  
        {  
            return ERROR;  
        }  
    }  
    else  
    {  
        restartStatus = RESTART;  
        bfRestartStack->unBundle  
            (batchControlInfo->GetBatchContextBuffer())  
    }  
    retVal = startingBatchFunction( this, startFuncContext,  
                                    startFuncContext );  
    if( retVal == SUCCESS )  
    {  
        retVal = Commit( FORCED );  
        if( retVal != SUCCESS )  
        {  
            return ERROR;  
        }  
  
        batchControlInfo->Delete();  
        return SUCCESS;  
    }  
    return ERROR;  
}
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 8

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
// default Commit() implementation supplied by framework
Virtual ErrorStatus Commit( int flag = UNFORCED )
{
    count++;
    if( flag == FORCED || count > commitThreshold )
    {
        BatchControlInfo *bcInfo =
            new BatchControlInfo( batchPgmId );
        long size = bfContextStack->sizeofClass();
        char *buf = new char [size];
        bfContextStack->bundle( buf );
        bcInfo->SetBatchContextBuffer( buf );
        bcInfo->SetRestartStatus( RESTART );
        status = bcInfo->Modify();
        if( status == ERROR )
        {
            return ERROR;
        }
        status = MC_Commit();
        if( status == ERROR )
        {
            return ERROR;
        }
        ResetCommitInfo();
        return SUCCESS;
    }
    return SUCCESS;
}

// default ResetCommitInfo() implementation supplied by framework
Virtual void ResetCommitInfo ()
{
    count = 0;

    int nElem = sizeof ( batchFunctionArray )
                / sizeof ( BatchFunctionDescriptor );
    for( int I = 0; I < nElem; I++ )
    {
        BatchFunctionDescriptor *elem =batchFunctionArray[I];
        BatchFunction *bf = elem->bfCommit;
        Bf->ResetCommitInfo( this );
    }
}
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)
NO. : /MUM/01

NO. OF SHEETS : 20
SHEET NO. : 9


PROVISIONAL SPECIFICATION

```
// default LogErrors() implementation supplied by framework
Virtual void LogErrors()
{
    Message *elem;
    For(int I=0, nElem=messageArray->nElem(); I < nElem; I++)
    {
        elem = (*messageArray)[nElem];
        elem->LogErrors();
    }
}

ErrorStatus GetMyContext(int &lbl, char *bfName, CppObj *&cntxt )
{
    int nElem = bfRestartStack->nElem();

    if( nElem == 0 )
    {
        return EMPTY;
    }
    for(int I = 0; I < nElem; I++)
    {
        BFStackFrame *elem;
        Elem = (BFStackFrame *)((*bfRestartStack)[I]);
        If( !strcmp( batchFunctionName, elem->GetName()) )
        {
            if( cntxt->IsDeepCopyReqd() )
            {
                cntxt->DeepCopy(elem->GetContext(),
                                cntxt);
            }
            else
            {
                cntxt = elem->GetContext();
            }
            lbl = elem->GetLabel();
            bfRestartStack->Delete( elem );
            return SUCCESS;
        }
    }
    return ERROR;
}
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 10

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
ErrorStatus PushContext(int label, char *batchFunctionName,
                        CppObj *batchFunctionContext )
{
    BFStackFrame *elem =
    new BFStackFrame( label, batchFunctionName,
                      batchFuctionContext );
    BfContextStack->Append( elem );
}

ErrorStatus PopContext( int& lbl, CppObj& batchFunctionContext )
{
    BFStackFrame *elem;
    Int nElem = bfContextStack->nElem();

    Elem = (*BTStackFrame *)((*bfContextStack)[nElem - 1]);
    BatchFunctionContext = elem->GetContext();
    Lbl = elem->GetLabel();
    BfContextStack->Delete( elem );
}

BatchFunctionDescriptor *SearchBatchFunction(char* bfName )
{
    // Performs binary search over batchFunctionArray[]
    // to return the array element
    // Array initialisation code should be generated
    // from the model so as to have batchFunctionNames
    // in the sorted order
}

ErrorStatus CallBatchFunction(char *bfName, CppObj *cntxt );
};

Class BatchControlInfo
{
private:
    char *batchPgmId;
    Long restartStatus;
    Void *batchContextBuffer;

Public:
    // Default constructor
    BatchControlInfo( char *pgmId )
    {
        batchPgmId = pgmId;
        batchContextBuffer = NULL;
    }
}
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 11

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
void *GetBatchContextBuffer() { return batchContextBuffer; }
void SetBatchContextBuffer( void *buf )
{
    batchContextBuffer = buf;
}

void *GetBatchPgmId() { return batchPgmId; }
void SetBatchPgmId( char *id )
{
    batchPgmId = id;
}

void *GetRestartStatus() { return restartStatus; }
void SetRestartStatus( long val )
{
    restartStatus = val;
}

// Std DM Create() method
ErrorStatus Create() {}

// Std DM Get() method
ErrorStatus Get() {}


// Std DM Modify() method
ErrorStatus Modify() {}

// Std DM Delete() method
ErrorStatus Delete() {}
};

class BFStackFrame : CppObj
{
private:
    char *name;
    CppObj *context;
    int label; // enum of PRE, POST and PROCESS

Public:
    BFStackFrame( int lbl, char *nm, CppObj *o ) :
        label(lbl), name(nm), context(o) {}
    Char *GetName() { return name; }
    CppObj *GetContext() { return context; }
    Void SetLabel( int val ) { label = val; }
    Int GetLabel() { return label; }
}
```

Fig - 5 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 12

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
Class BatchFunction
{
protected:
    long count;
    long commitThreshold;
public:
    virtual char *GetBatchFunctionName() { return ""; }

    virtual ErrorStatus Commit( BatchProgram *bp )
    {
        BatchFunctionDescriptor *bpDesc =
            bp->SearchBatchFunction( GetBatchFunctionName() );

        BatchFunction *singleton = bpDesc->bfCommit;
        Singleton->count++;
        If(singleton->commitThreshold != 0 )
        {
            If( singleton->count > singleton->commitThreshold )
            {
                return bp->Commit( FORCED );
            }
        }
        bp->Commit();
    }

    // default implementation provided by framework
    virtual void ResetCommitInfo(BatchPgm *bp)
    {
        count = 0;
    }
};

Struct BatchFunctionDescriptor
{
    char *batchFunctionName;
    BatchFn batchFunction;
    Bool memMgmtReqd;
    BatchFunction *bfCommit;
}

Struct CommitControl
{
    char *batchFunctionName;
    void *commitData;
}
```

Figure 5



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 13

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
Class <BatchProgramName> : BatchPgm
{
public:
    <BatchProgram> ( char *batchProgramId, BatchFn startFunc ) :
        BatchPgm( batchProgramId, startFunc ) {}

    ErrorStatus Init( int argc, char* argv )
    {
        // default behaviour provided by framework can be
        // overridden to set startFuncContext as required
        return BatchPgm::Init( srgc, argv );
    }
    ErrorStatus Commit()
    {
        // default behaviour provided by framework can be
        // overridden as required
        return BatchPgm::Commit();
    }
    // This function overrides ResetCommitInfo()
    ErrorStatus ResetCommitInfo()
    {
        BatchPgm::ResetCommitInfo();
        // Codes the overriding behaviour here
    }

    void LogErrors()
    {
        // default behaviour provided by framework can be
        // overridden as required
        return BatchPgm::LogErrors();
    }
};

static <BatchProgramName>::batchFunctionArray[] =
{
    { "<batchFunctionName>", <BatchFunctionName>::sh_<batchFunctionName>,
      TRUE / FALSE, NULL },
    :
    :
    :
};
```

Figure 6



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 14

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
Int main( int argc, char *argv)
{
    ErrorStatus retVal;

    // Instantiate the modelled batch program
    <BatchProgramName> *batchProgram =
    new <BatchProgramName>(<BatchProgramName>,<startBatchFunction>);

    retVal = batchProgram->Setup( argc, argv );
    if( retVal == ERROR )
    {
        batchProgram->LogErrors();
        exit(1);
    }
    retVal = batchProgram->Exec();
    if( retVal == ERROR )
    {
        batchProgram->LogErrors();
        exit(1);
    }
    retVal = batchProgram->Cleanup();
    if( retVal == ERROR )
    {
        batchProgram->LogErrors();
        exit(1);
    }
    exit(0);
}
```

Figure 7



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET. NO. : 15

NO. : /MUM / 01

PROVISIONAL SPECIFICATION

```
Class <BatchFunctionName> : BatchFunction
{
public:
    // Pre, Process and Post methods are implemented by user
    ErrorStatus Pre( BatchPgm *batchProgram, <Context> *inContext,
<Context> *& outContext );
    ErrorStatus Process( BatchPgm *batchProgram, <Context>
*inContext, <ReturnTypeOfDataProcessor> *inData, <Context> *&outContext
);
    ErrorStatus Post( BatchPgm *batchProgram, <Context> *inContext,
<Context> *& outContext );

    // sh_<batchFunctionName> is generated
    ErrorStatus sh_<batchFunctionName>( BatchPgm *batchProgram,
<Context> *inContext, <Context> *& outContext );

    Char *GetBatchFunctionName() { return "<BatchFunctionName>"; }

    // This function overrides commit()
    ErrorStatus Commit( BatchPgm *bp )
    {
        BatchFunction::Commit( bp );
        // Codes the overriding behaviour here
    }

    // This function overrides ResetCommitInfo( BatchPgm *bp )
    ErrorStatus ResetCommitInfo( BatchPgm *bp )
    {
        BatchFunction::ResetCommitInfo(bp);
        // Codes the overriding behaviour here
    }
};
```

Figure 8



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 16

NO. : / MUM / 01

PROVISIONAL SPECIFICATION

```
ErrorStatus <BatchFunctionName> :: sh_bfl( BPl *batchProgram,
    Conl *inContext, Conl *& outContext )
{
    ErrorStatus status;
    Int lbl;
    Long restartStatus = batchProgram->GetRestartStatus();
    Conl *context = ( inContext == NULL ) ? new Conl : inContext;

    If( restartStatus == RESTART )
    {
        status = batchProgram.GetMyContext(lbl, "bfl", context );
        if( status != EMPTY )
        {
            if( status != SUCCESS )
            {
                return ERROR;
            }
            switch( lbl )
            {
                case PRE: goto Pre_label;
                case PROCESS: goto Process_label;
                case POST: goto Post_label;
            }
        }
    }

    Pre_label:
    Status = batchProgram.PushContext( PRE, "bfl", context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
    Status = Pre_bfl( batchProgram, context, context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
    Status = batchProgram.PopContext( lbl, context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
}
```

Figure 9 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)
NO. : /MUM / 01

NO. OF SHEETS : 20
SHEET NO. : 17

PROVISIONAL SPECIFICATION

```
Process label:
// Return value of data provider of bfl
DP1_out *dataObj = new DP1_out;

Status = DP1::Open( context );
if( status != SUCCESS )
{
    return ERROR;
}

While( 1 )
{
    status = DP1::Fetch( dataObj );
    if( status == ERROR )
    {
        return ERROR;
    }
    if( status == DATA_OVER ) break;

    Status=batchProgram.PushContext(PROCESS,"bfl", context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
    status=Process_bfl( batchProgram, context,
        dataObj, context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
    status = Commit( batchProgram );
    if( status != SUCCESS )
    {
        return ERROR;
    }

    Status = batchProgram.PopContext( lbl, context );
    if( status != SUCCESS )
    {
        return ERROR;
    }
}
```

Figure 9 (continued)



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 18

NO. : / MUM / 01

PROVISIONAL SPECIFICATION

```
DP1::Close();

Post_label:
Status = batchProgram.PushContext( POST, "bfl", context );
if( status != SUCCESS )
{
    return ERROR;
}
Status = Post_bfl( batchProgram, context, context );
if( status != SUCCESS )
{
    return ERROR;
}
Status = batchProgram.PopContext( lbl, context );
if( status != SUCCESS )
{
    return ERROR;
}
outContext = context;
return SUCCESS;
```

Figure 9



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20
SHEET NO. : 19

NO. : /MUM/01

PROVISIONAL SPECIFICATION

```
ErrorStatus BatchPgm::CallBatchFunction(char *bfName, CppObj *cntxt )
{
    long PrevId, poolId;
    BatchFunctionDescriptor *bfDescriptor;
    ErrorStatus status;

    BfDescriptor = SearchBatchFunction( bfName );
    If( bfDescriptor == NULL )
    {
        return ERROR;
    }

    if( bfDescriptor->memMgmtReqd == TRUE )
    {
        prevId = GetCurPoolId();
        poolId = CreatePool();
        SetCurPool( poolId );
        If( cntxt != NULL )
        {
            cntxt.SetDeepCopy( TRUE );
        }
    }

    status = bfDescriptor->batchFunction( this, cntxt );
    if( status != SUCCESS )
    {
        return ERROR;
    }

    if( bfDescriptor->memMgmtReqd == TRUE )
    {
        SetCurPool( prevId );
        FreePool( poolId );
    }
}
```

Figure 10



MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY

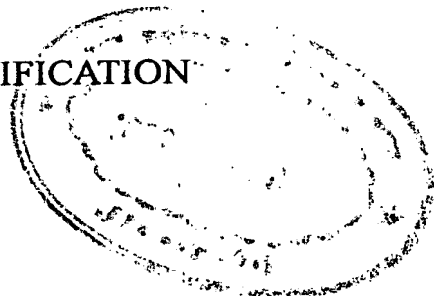
NAME : TATA CONSULTANCY SERVICES
(TATA SONS LIMITED)

NO. OF SHEETS : 20

SHEET NO. : 20

NO. : /MUM/01

PROVISIONAL SPECIFICATION



```
Class CppObj
{
private:
    ErrorStatus deepCopyFlag;

public:
    void SetDeepCopy( ErrorStatus value ) { deepCopyFlag = value; }
    ErrorStatus IsDeepCopyReqd() { return deepCopyFlag; }
    ErrorStatus DeepCopy( CppObj *src, CppObj *dest )
    {
        long size = src->sizeofClass();
        char *buf = new char [ size ];
        src->bundle( buf );
        dest->unbundle( buf );
        return SUCCESS;
    }
}
```

Figure 11

```
<modelledClass> :: Set<attributeName>( <modelledClassMember> *src )
{
    if( IsDeepCopyReqd() )
    {
        DeepCopy( src, this-><attributeName> );
    }
    else
    {
        <attributeName> = src;
    }
}
```

Figure 12

MOHAN DEWAN
APPLICANT'S PATENT ATTORNEY